John von Neumann Institute for Computing

Performance Measurements and Analysis of the BlueGene/L MPI Implementation

Michael Hofmann, Gudula Rünger

published in

Parallel Computing: Architectures, Algorithms and Applications ,
C. Bischof, M. Bücker, P. Gibbon, G.R. Joubert, T. Lippert, B. Mohr, F. Peters (Eds.),
John von Neumann Institute for Computing, Jülich,
NIC Series, Vol. **38**, ISBN 978-3-9810843-4-4, pp. 405-412, 2007.
Reprinted in: Advances in Parallel Computing, Volume **15**,
ISSN 0927-5452, ISBN 978-1-58603-796-3 (IOS Press), 2008.

http://www.fz-juelich.de/nic-series/volume38

Performance Measurements and Analysis of the BlueGene/L MPI Implementation

Michael Hofmann[a] **and Gudula Rünger**

Department of Computer Science
Chemnitz University of Technology
E-mail: {mhofma, ruenger}@informatik.tu-chemnitz.de

The massively parallel architecture of the BlueGene/L supercomputer poses a challenge to the efficiency of parallel applications in scientific computing. The specifics of the BlueGene/L communication networks have a strong effect on the performance of the MPI implementation. Various optimizations and specially adapted algorithms cause varying performance results for communication operations. In this paper, we present performance results of various MPI operations for a BlueGene/L system using up to 4,096 nodes. We discuss their efficiency in terms of the communication hardware and investigate influences on the performance of commonly used point-to-point and collective operations. The results give an overview of the efficient usage of the MPI operations and can be used to optimize the performance of parallel applications.

1 Introduction

During the last few years, one important progress in high performance computing was the development and deployment of the IBM BlueGene/L (BG/L) supercomputer. This was an important milestone of the ongoing effort to build a petaflop supercomputer that provides sufficient computing power for advanced scientific computations. Beside the superior theoretical peak performance of about 360 teraflops, the most interesting and also challenging properties arise from the unique architecture. The BG/L is a massively parallel distributed memory system with several special purpose networks. Message passing programming is supported by an MPI implementation especially adapted to the properties of the communication networks. Achieving high performance in parallel scientific applications requires an adaption to the target environment. This involves the communication infrastructure given through the MPI implementation. Knowledge about their efficient usage is essential for programmers to prepare their applications to scale well up to thousands of nodes. The properties of the BG/L system have a direct influence on the performance of MPI communication operations. Due to specific optimizations, the performance of MPI operations varies depending on their particular usage. In this paper, we present performance results of MPI operations for a BG/L system[b] using up to 4,096 nodes. We discuss their origins and derive implications for an efficient usage.

The rest of this paper is organized as follows. In the following, we introduce the BG/L system and list related work. Section 4 presents performance results of MPI point-to-point communication operations and Section 5 shows results of collective MPI operations. Sections 6 presents results for communication schemes like overlapping communication and nearest neighbour communication. We conclude in Section 7.

[a] Supported by Deutsche Forschungsgemeinschaft (DFG)
[a] Supported by Deutsche Forschungsgemeinschaft (DFG)
[b] Measurements are performed on the BlueGene/L system at the John von Neumann Institute for Computing, Jülich, Germany. `http://www.fz-juelich.de/zam/ibm-bgl`

2 BlueGene/L System

The BlueGene/L supercomputer consists of up to 65,536 (64Ki[c]) dual-processor compute nodes with 700 MHz PowerPC based processors and either 512 MiB or 1 GiB main memory. Two operation modes are available for utilizing the dual-processors nodes: In *co-processor mode* (CO) each node runs only one process and allows the second processor to be used as a communication or computation coprocessor. In *virtual node mode* (VN) two separate processes run on every node with evenly divided resources. (The following investigations use CO mode unless otherwise stated.) The system features several communication networks, three of those available for message passing communication. The *global interrupt* network can be used to perform a barrier synchronization on a full system in 1.5 μs. The *collective* network is a tree with fixed point arithmetic support in every node and can be used to perform broadcast and reduction operations. For reduction operations with floating-point data, a two-phase algorithm consisting of multiple fixed point reduction operations was introduced[1]. The collective network has a payload bandwidth of about 337 MB/s[c] and a latency of 2.5 μs on a 64Ki nodes system. For point-to-point communication, a three dimensional *torus* network connects every node to six neighbors through bi-directional links. Data is sent packet-wise with packet sizes up to 256 bytes and the maximum payload bandwidth per link and direction is about 154 MB/s (we refer to this as the single link bandwidth). The *deposit bit* feature of the torus network supports fast broadcasting of packets to a line of nodes. A 64Ki nodes system is connected by a 64x32x32 torus network and can be split into multiple independent partitions. The BG/L MPI implementation[2] is based on MPICH2 and especially adapted to make use of the different communication networks.

3 Related Work and Motivation

The BlueGene/L has been introduced in detail[3] and a lot of information about application development and tuning is available[4]. Details about the BG/L MPI implementation[2] include information about different communication layers as well as performance results of single MPI operations and parallel application benchmarks. The collective MPI operations have been extensively optimized[1] to make use of the specialties of the communication networks. Separate algorithms for short and long message communication are introduced and performance results for varying message sizes are shown. The implementation of a one-sided communication interface[5] enables the usage of global address space programming models and one-sided operations of MPI-2 and presents additional efforts for bandwidth and latency optimizations. The topology functions of MPI have also been optimized[6] to improve the mapping of virtual (application specific) topologies to the physical topology of the torus network. MPI performance analysis tools were ported to the BG/L system and have been tested using a number of scientific applications[7].

Most of the work about the BG/L MPI focuses on implementation details for certain MPI operations and optimized algorithms for these implementations. In contrast, this article concentrates on the performance of MPI operations when used within a parallel appli-

[c]Prefixes Ki, Mi, and Gi are used to denote the base-1024 versions of Kilo, Mega, and Giga. Prefixes K, M, and G represent the base-1000 versions.

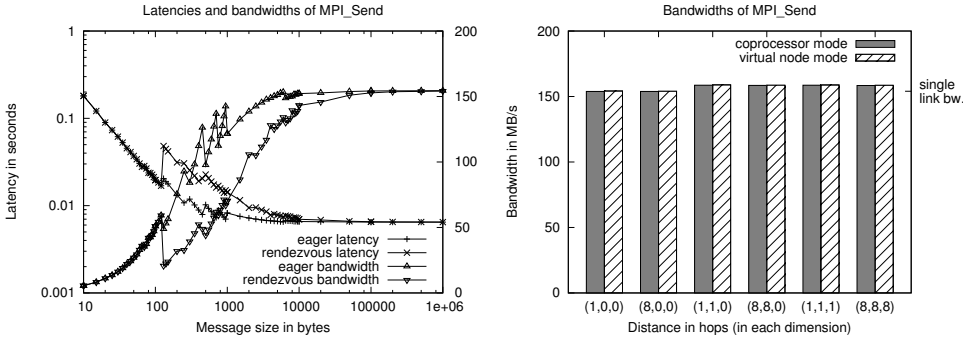

Figure 1. Latencies and bandwidths of point-to-point communication depending on the message size (left) and on the distance to the target node (right).

cation. The detailed measurements show how specific implementations of an MPI operation have an effect on the application program. Various influences on the performance of MPI operations are investigated and it is shown how certain communication schemes of an application program can be implemented best using the underlying MPI library.

4 MPI Point-to-Point Communication

All MPI point-to-point communication uses the torus network with either *deterministic* routing (packets use fixed paths) or *adaptive* routing (path depends on the current load). Depending on the size of the message to be sent, one of three protocols is chosen. The *eager* protocol is optimized for latency and used for medium size messages. The sender immediately starts sending the packets assuming that the receiver is able to handle them. The *short* (or *one packet*) protocol is used for messages that can be sent with one packet and causes a very low overhead. Both protocols use deterministic routing. The *rendezvous* protocol is a bandwidth optimized protocol for large messages. It uses a handshake mechanism between sender and receiver to establish a message context. Adaptive routing is used to maintain a balanced network load. The usage of the eager or the rendezvous protocol can be controlled with the `BGLMPI_EAGER` variable.

Figure 1 (left) shows the latencies and the bandwidths for sending 1 MB data with `MPI_Send` and varying message sizes to a neighboring node. The usage of the eager and the rendezvous protocol is forced using the `BGLMPI_EAGER` variable. Up to a message size of about 120 bytes the short protocol is used. In this case, with an increasing message size the total number of packets decreases resulting in constantly decreasing latencies. For message sizes of about 120-130 bytes the switch to the eager or the rendezvous protocol occurs. There is a clear difference between the two protocols with the eager protocol being about two times faster. For both protocols, there is a zigzag scheme visible for message sizes up to 1000 bytes. The peaks occur when an additional packet is required to send messages of the particular size. With increasing message sizes, the differences between the two protocols vanish and their latencies become fairly equal and constant for messages sizes above 10000 bytes. The short protocol achieves about 38 % of the single link bandwidth, while the eager and the rendezvous protocol reach about 99 %.

Figure 1 (right) shows the bandwidths for sending a message of size 1 MB to nodes at different distances in a 16x16x16 partition. It can be observed that the distance has no effect on the achievable bandwidth, because distances (1,0,0) and (8,0,0) show the same results. A slight increase occurs if sender and receiver nodes are not located on the same axis. However, the results indicate that only one link is used for a point-to-point operation at a time, since the bandwidth does not clearly exceed the single link bandwidth. Communication between processes on the same node in VN mode is done with memory copies and achieves a bandwidth of about 1670 MB/s. Apart from that, the operation mode has no effect on the results presented above.

The results confirm that the performance of the point-to-point operations strongly depends on the message size and the communication protocol used. The eager protocol shows generally the best performance and has significant performance peaks for certain message sizes. Comparable performance with the rendezvous protocol requires message sizes of at least 10000 bytes. However, with a higher network load, the rendezvous protocol can become more important. The distance between sender and receiver has no significant effect on the bandwidth of single communication operations and efforts for improving the locality are unnecessary in this case. The bandwidth of a single point-to-point operation is limited to the bandwidth of a single torus link and multiple link performance as demonstrated for one-sided communication[5] is currently not achievable.

5 MPI Collective Communication

For efficient implementations of MPI collective communication the BG/L system has two special purpose networks, the global interrupt network and the collective network. However, the usage of these networks is limited to operations on `MPI_COMM_WORLD`. Otherwise, the torus network is used with a number of optimized algorithms[1]. Especially for rectangular subsets of nodes optimizations based on the deposit bit feature are used.

MPI_Bcast / MPI_Allreduce

The following results of broadcast and reduction operations are obtained using a fixed message size of 1 MB. Figure 2 (top) shows latencies of `MPI_Bcast` and `MPI_Allreduce` (with `MPI_SUM`) for varying numbers of nodes using the CO mode. For `MPI_COMM_WORLD` the broadcast and the reduction operation with integer data have almost identical results independent from the number of nodes. The latency of about 2.98 ms for 1 MB data corresponds to a bandwidth of about 336 MB/s which is close to the payload bandwidth of the collective network. The latencies for floating-point data reductions are about 3-4 times higher, because of the missing arithmetic support in the collective network. With subsets of nodes, the torus network is used and optimized algorithms based on the deposit bit feature of the torus network are applied for rectangular subsets. With a 16x16x16 partition the broadcast operation achieves full single link bandwidth for broadcasting in an incomplete line (2-15 nodes), full double link bandwidth for broadcasting in a full line (16x1x1) or incomplete plane (16x2x1, ..., 16x15x1), and exceeds the maximum bandwidth of the collective network for broadcasting in a full plane (16x16x1) or cubic subset (16x16x2, ..., 16x16x16). The results of `MPI_Allreduce` show similar optimizations for rectangular subsets, but without achieving maximum bandwidths and

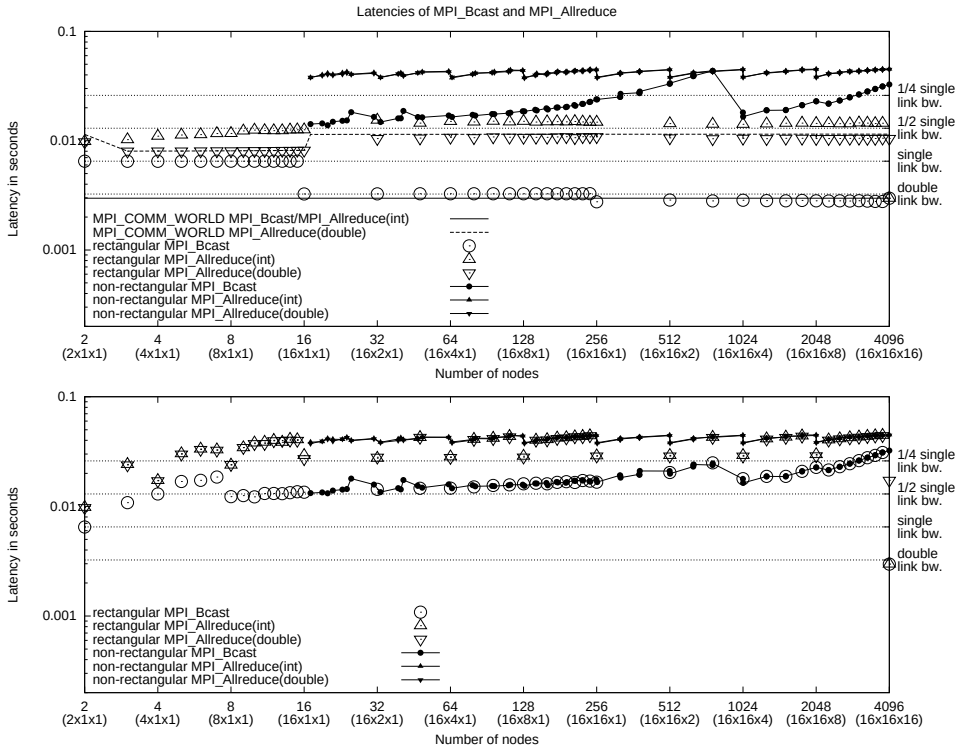

Figure 2. Latencies of MPI broadcast and reduction operations depending on the number of participating nodes and the shape (rectangular or non-rectangular) of the subset of nodes in CO (top) and VN (bottom) mode.

independent from the number of utilized links. With non-rectangular subsets of nodes the latencies are at least 3-4 times higher. Figure 2 (bottom) shows results using the VN mode. The improvements due to the optimizations for rectangular subsets of nodes have almost disappeared. Only for certain rectangular subsets of nodes there is a small decrease in latency for the reduction operation. For the broadcast operation there is no difference between rectangular and non-rectangular subsets.

Figure 3 (left) shows latencies of `MPI_Allreduce` depending on the reduction operation and the data type in three different situations (`MPI_COMM_WORLD`, rectangular and non-rectangular subsets). A user-defined reduction operation (summation) disables the optimizations and achieves almost equal results in all three situations. The results for the non-rectangular subset are independent from the pre-defined reduction operation and the data type. For `MPI_COMM_WORLD` the collective network can be used, but the maximum performance is only achieved with integer data and `MPI_SUM`. With `MPI_PROD` the latency increases by an order of magnitude. The results with floating-point data are the same for the pre-defined reduction operations and fairly independent from the number of nodes.

The results show that the performance of the collective operations strongly depend on the optimizations due to the BG/L communication hardware. The actual number of participating nodes has only a small influence, thus indicating good scalability of the collective

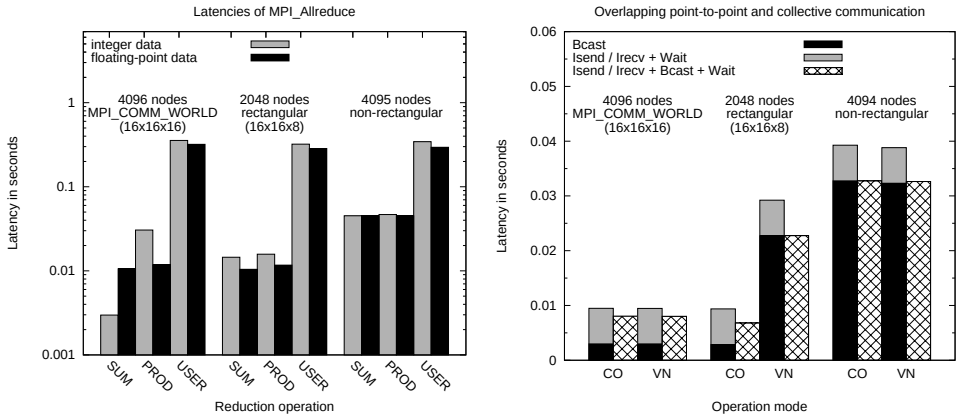

Figure 3. Latencies of `MPI_Allreduce` with different reduction operations and data types. (left) Latencies for overlapping point-to-point and collective MPI operations (`MPI_Bcast`). (right)

operations. Non-rectangular subsets of nodes prevent the usage of optimized algorithms, resulting in a significant performance loss. Therefore, it is almost better to increase the number of nodes participating in a collective operation to gain a rectangular shape. Additionally, it is recommend to spread subsets of nodes at least across two dimensions of the torus network to benefit from the usage of multiple torus links.

6 Communication Schemes

Overlapping Communication

The non-blocking communication operations of MPI can be used to overlap communication and computation. This is a common technique for hiding latencies of communication operations. However, with the current BG/L MPI version this kind of processing has no benefit. Overlapping point-to-point communication with a single matrix multiplication task produces exactly the same latency results as if they were performed one after another. This behaviour was the same using CO and VN mode. Figure 3 (right) shows measurements for overlapping point-to-point communication with a collective operation. Pairs of nodes exchange 1 MB messages with non-blocking operations while they are participating in a broadcast of a 1 MB message. Results are shown for three situations (`MPI_COMM_WORLD`, rectangular and non-rectangular subsets) with different broadcast implementations in use. Overlapping the two communication operations leads to a decrease of the latencies of about 15 % to 25 %. Significant differences between CO and VN mode occur only with the rectangular subset of nodes. In this case, the VN mode leads to an increased latency of the broadcast operation (as already show in Section 5). However, the improvements due to the overlapping are the same using CO and VN mode.

Nearest Neighbour Communication

Figure 4 (left) shows the bandwidths of concurrent communication of one node with its six direct neighboring nodes. Results are shown for receiving, sending, and exchang-

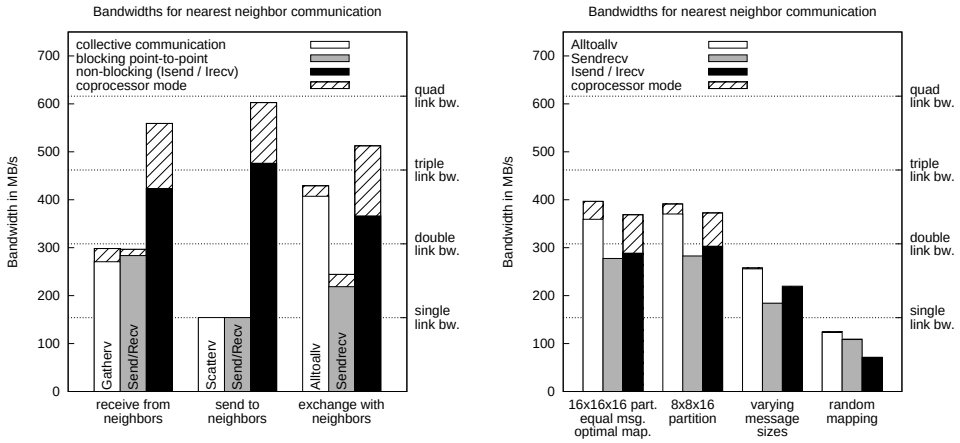

Figure 4. Bandwidths of one node communicating with its six neighboring nodes in the torus network using different MPI operations. (left) Per node bandwidths if all nodes communicate with their neighbors in a 16x16x16 torus network using different MPI operations. (right)

ing (concurrent send and receive) messages of size 1 MB with each neighbour using different communication operations (collective, blocking, and non-blocking point-to-point). When only sending or receiving, the differences between collective (`MPI_Gatherv` or `MPI_Scatterv`) and blocking point-to-point communication are rather small. Sending data to the neighbors achieves single link bandwidth, while data from the neighbors is received with about double link bandwidth. Higher rates are achieved when performing all communication at once with non-blocking operations. For exchanging data, the bandwidth of the collective operation (`MPI_Alltoallv`) is much better than with consecutive blocking point-to-point operations. The usage of `MPI_Sendrecv` achieves only about 80 % of the maximum bi-directional performance. With the CO mode, the usage of non-blocking operations achieves the best results, while with the VN mode the collective operation performs best. In general, the bandwidths of the non-blocking operations with the CO mode are about 30 % higher in comparison to the VN mode. The benefits for the collective and the blocking operations are rather small.

Figure 4 (right) shows the bandwidths per node if all nodes communicate with their neighbors. With a 16x16x16 partition, equal message sizes (1 MB) and an optimal mapping (communication with direct neighbors in the torus network only) the usage of `MPI_Alltoallv` achieves the best result of about 400 MB/s. The bandwidth with non-blocking point-to-point operations is only slightly lower and `MPI_Sendrecv` achieves again only about 80 % of the maximum bi-directional bandwidth. With a smaller partiton (8x8x16) the bandwidths for all communication operations remain fairly unchanged. Using messages with varying sizes (0.5-1.5 MB) leads to a decrease of the bandwidths of about 30-40 %. A random mapping of the nodes in the torus network decreases the locality of the communication significantly. Therefore, the bandwidths for all communication operations fall below the single link bandwidth and the non-blocking operations become the worst. Using the CO mode improves the performance of collective and non-blocking operations, but only under optimal conditions (equal message sizes, optimal mapping).

Even though the non-blocking operations of the BG/L MPI cannot be used to overlap communication and computation, they still can be used to perform multiple communication operations at once. Performance benefits can be achieved from the concurrent usage of the collective and the torus network as well as from the utilization of multiple torus links. The usage of all-to-all operations for data exchange with neighbors in a grid achieves similar good performance like a direct implementation with point-to-point operations. With non-optimal conditions like varying message sizes or decreased locality of the communication, the usage of all-to-all communication performs best. Using the CO mode increases the performance, especially for performing multiple non-blocking operations at once.

7 Conclusion

The measurements have shown that the specifics of the BG/L communication hardware have significant effects on the performance of single MPI operations. Optimized algorithms make of use the different communication networks and improve the performance of collective MPI operations. However, their usage is subject to a number of restrictions. Maximum performance in terms of hardware and software limitations is achieved for several communication operations and enables the prediction of latency values. Non-blocking operations have shown to be useful to communicate with various nodes at once and to utilize multiple links of the torus network. The all-to-all communication operation provided with the BG/L MPI turns out to be a good choice for exchanging data, especially under non-optimal conditions. In general, the experiments have shown that good scalability of communication operations can be achieved even for up to thousands of nodes.

References

1. G. Almási et al., *Optimization of MPI collective communication on BlueGene/L systems*, in: ICS '05: Proc. 19th annual international conference on Supercomputing, pp. 253–262, (2005).
2. G. Almási et al., *Design and implementation of message-passing services for the BlueGene/L supercomputer* , IBM J. of Research and Development, **49**, no. 2/3, (IBM, 2005).
3. G. L.-T. Chiu, M. Gupta, and A. K. Royyuru, (Eds.), *IBM J. of Research and Development: Blue Gene* , **49**, no. 2/3, (IBM, 2005).
4. O. Lascu et al., *Unfolding the IBM eServer Blue Gene solution* , IBM Redbooks, (2005).
5. M. Blocksome et al., *Design and implementation of a one-sided communication interface for the IBM eServer Blue Gene supercomputer*, in: SC '06: Proc. 2006 ACM/IEEE Conference on Supercomputing, p. 120, (2006).
6. H. Yu, I-H. Chung and J. Moreira, *Topology mapping for Blue Gene/L supercomputer*, in: SC '06: Proc. 2006 ACM/IEEE conference on Supercomputing, p. 116, (2006).
7. I.-H. Chung, R. E. Walkup, H.-F. Wen and H. Yu, *MPI performance analysis tools on Blue Gene/L*, in: SC '06: Proc. 2006 ACM/IEEE conference on Supercomputing, p. 123, (2006).